Model-based Decentralized Embedded Diagnosis inside Vehicles: Application to Smart Distance Keeping Function

Othman Nasri, Hassan Shraïm, Philippe Dague, Olivier Héron, Mickael Cartron

Model-based Decentralized Embedded Diagnosis inside Vehicles: Application to Smart Distance Keeping Function

Othman Nasri*, Hassan Shraim*, Philippe Dague*, Olivier Héron† and Mickael Cartron†
*LRI, Université Paris-Sud 11, CNRS & INRIA Saclay - Île-de-France
Bât 490, 91405 Orsay Cedex France
Emails: othman.nasri@lri.fr, hassan.shraim@gmail.com, philippe.dague@lri.fr
†CEA LIST, Saclay
91191 Gif-sur-Yvette cedex France
Emails: olivier.heron@cea.fr, mickael.cartron@cea.fr

***Abstract*—In this paper, the deployment of a fault diagnosis strategy in the Smart Distance Keeping (SDK) system with a decentralized architecture is presented. The SDK system is an advanced version of the Adaptive Cruise Control (ACC) system, implemented in a Renault-Volvo Trucks vehicle. The main goal of this work is to analyze measurements, issued from the SDK elements, in order to detect, to localize and to identify some faults that may be produced. Our main contribution is the proposition of a decentralized approach permitting to carry out an on-line diagnosis without computing the global model and to deploy it on several control units. This paper explains the model-based decentralized solution and its application to the embedded diagnosis of the SDK system inside truck with five control units connected via a CAN-bus using "Hardware In the Loop" (HIL) technique. We also discuss the constraints that must be fulfilled.**

***Keywords*-Smart Distance Keeping (SDK), vehicle modeling, model-based embedded diagnosis, decentralized architecture, Hardware in the loop (HIL).**

I. INTRODUCTION

In order to respond to the increasing demands of safety and driving comfort, more and more electronic functions are embedded in the vehicles such as engine control (to optimize fuel economy and to reduce pollution), ABS (Antilock Braking System), ESP (Electronic Stability Program), etc. Each global safety or comfort system contains one or more functions which may be distributed on several ECUs (Electronic Control Unit). Most of these functions are modular and respecting some norms (such as AUTOSAR). They exchange information (for example vehicle speed) with other functions via communication interfaces. That means, if the system is not equipped with a certain diagnosis strategy, any fault generated from a function, may influence all the functions with are related to it. This fact highlights the problem of fault propagation and the need of fault diagnosis in the vehicle.

In this paper, we focus on the sensors fault diagnosis of the Smart Distance Keeping (SDK) system, which is an advanced Adaptive Cruise Control (ACC) system implemented in a Renault-Volvo Trucks vehicle to increase safety by overcoming some ACC limitations [1]. It is composed of many subsystems (micro controllers, cables, CAN or FlexRay bus, sensors, actuators, etc.) coming from different suppliers. That is why, its diagnostic is a challenging task.

Considering the size and the complexity of the SDK system, a centralized on-board diagnosis is not adequate because it requires the establishment of a global model of the system and too much memory resources and prevents to act immediately each time a diagnosis could be find at local level. To detect and isolate possible faults in the SDK system and manage its architectural complexity, we have chosen the model-based fault diagnosis approach (FDI) in a decentralized manner. A local diagnoser is associated with each component of the SDK system based on a modular modeling of the plant elements. All local diagnosis decisions are transmitted via CAN-bus and merged by a dedicated supervisor in order to obtain a global decision and carry out any recovery action. A strategy for applying this merging operation was developed in order to be efficient.

The paper is organized as follows. In section II, we present the decentralized model-based fault diagnosis approach. Section III explains how to deploy this approach in order to apply on-line diagnosis for the SDK system. Then, we evaluate the proposed diagnosis approach in section IV. Finally, we conclude with a discussion on the related work.

II. DECENTRALIZED DIAGNOSIS

In most automated systems, the command part (which implements the control of the operative part) is generally represented through a model to be applied to the operative part (mechanical components which should be controlled by means of actuators, such as engines). Realizing a diagnosis requires also to be able to represent the state of the operative part using a model that can be integrated to the one of the command part, separated or mixed. Thus, when a fault occurs, it is possible to get information regarding the process and to compare model and process. This is called model-based diagnosis (Fault Detection and Isolation) [2].

The method of fault detection and isolation (FDI) is based on the use of analytical redundancy (model-based), i.e. relations among the measured variables (see figure 1). It can be divided into several steps [3].

- First, data containing information about the process states (i.e., the symptoms) are transmitted to the residual generation module. This module generates a vector that carries information about a particular fault.
- Second, the generated vectors are evaluated and filtered, i.e. faults are localized and identified, in order to extract the primary cause of the observed evolution. The structured residual approach [4] has been chosen in order to perform this stage.

Figure 1. Model-based fault diagnosis using residual approach

A. Motivations

A decision-making structure for fault diagnosis must be defined to face combinatorial explosion and real time problems and/or communication problems between various components of a process. The choice of a structure depends on the distribution of the available information (model and observation): centralized or distributed, and on the nature of the process: simple with only one control unit or complex with several local control units. There are therefore three main structures of decision-making methods for diagnosis: centralized, decentralized and distributed.

The centralized structure consists in associating to one global model of the process a single diagnostic module (called diagnoser). It collects the different process informations before making its final decision on the operating status of the process [7]. Although successful in terms of diagnosis for simple systems, the centralized structure is difficult to use for large systems. Indeed, the acquisition of a global model of the process rises difficulties and often leads to combinatorial explosion problem.

The decentralized structure is based on several local independent diagnosers that are associated to one global model of the process . Each diagnoser receives the observations which are specific to it and takes a local decision based on its local observations. However, this structure involves problems of indecision when some global specifications require consistency checking between decisions of local diagnosers. To solve these cases, each diagnoser sends its local decision to a coordinator (or supervisor) which will manage the different problems of ambiguity between these diagnosers and will take the final decision [8].

In the distributed structure, the process is modeled through its components (or subsystems) by several local models. Each one is equipped with a local diagnoser responsible of it. In the case of global specifications, a communication protocol allows directly the communication between the different diagnosers to manage conflict decision [9]. Each diagnoser makes its decision based on its own local observation and that reported by other local diagnosers as answers to its queries. This structure permits to throw off the construction of a coordinator but implies the definition of a protocol for decision making through communication between diagnosers, often impractical because without guarantee of convergence in bounded time and generating more important communication traffic and delays.

In this paper we adapt the decentralized/distributed structure. In other words, a local diagnoser is associated with each component of the process based on a modular modelling of the plant elements. All local diagnosis decisions are transmitted via a communication environment and merged by a dedicated supervisor in order to obtain a global decision and carry out any recovery action (see figure 2). This fusion can be realized by a coordinator based on a set of rules. The goal of this coordinator is to solve the problem of decision conflict and/or ambiguity among local diagnosers in order to obtain a diagnosis performance equivalent to that of a centralized diagnoser. This approach allows carrying

Figure 2. Model-based decentralized principle with 2 local diagnosers

out on-line diagnosis without computing the global model and overcoming both the combinatorial and communication traffic explosion problems.

B. SDK algorithm

Smart Distance Keeping (SDK) or ”enhanced Adaptive Cruise Control (ACC)” is a system which automatically controls the vehicle’s longitudinal position, by acting on the engine, gearbox, retarder and braking system. This requires the vehicle to be equipped with a radar system linked to a dedicated control unit as shown in figure 3. The global

Figure 3. Radar installed on the Renault Magnum truck

SDK system may be decomposed into two main parts, the SDK controller, and the SDK environment model. The main functions of the SDK controller are to: (i) acquire the distance between the truck and the front object, (ii) compute the deceleration (acceleration) needed to realize the correct functioning of the SDK system (maintaining a minimal safety distance with the front vehicle), and (iii) use a control algorithm for acting through engine, braking system, etc. in order to adjust the velocity of the truck. From the SDK architecture, we see that the decision of the SDK controller depends essentially on the data issued from some sensors (wheels angular velocity sensors, radar, and transmission sensor), which means that any faulty data will influence the SDK system decision. That is why, one local

diagnoser is associated with each one of these sensors in order to diagnose an SDK fault by using the decentralized model-based approach (see figure 2).

Figure 4. Decentralized model based fault diagnosis of the SDK system

C. Decentralized diagnosis of the SDK

A model of the SDK environment, i.e. the part of the vehicle required to close the loop, is necessary to perform diagnosis (see figure 2). So, by applying the laws of dynamics, a simplified model of the diesel engine has been developed (see [5] and [6] for more details). It permits to identify the angular velocities of the six wheels and the crank shaft angular velocity in response to an action (on braking system or accelerator pedal or gearbox).

1) Wheels diagnoser: The six wheels angular velocities w_i are the inputs of a local algorithm able to detect and isolate any fault that occurs in the wheels velocity sensors. Indeed, it is possible by using these redundant measurements to generate a set of structured residuals and afterwards detect and isolate single or multiple fault. The outputs of this algorithm are the state (normal/abnormal) s_i of the wheels sensors. In addition, it computes the longitudinal speed V_w of the truck which is approximated based on the non faulty sensors:

$$(s_1, s_2, s_3, s_4, s_5, s_6, V_w) = wheels(w_1, w_2, w_3, w_4, w_5, w_6)$$

2) Transmission diagnoser: Since there is no redundancy measurement, the algorithm just computes the longitudinal truck speed V_t by using the value of the crank shaft angular velocity w_e (the gear box output) and the transmission rate number n:

$$V_t = transmission(w_e, n)$$

3) Radar diagnoser: The radar provides the relative distance RD and velocity RV between the front object and the truck. To detect locally a radar fault, three scenarios have been analyzed:

- if the radar is faulty and does not detect any object then, strictly speaking, without the help of another device, we can do nothing, but in practice the symptom, permanently flat signals, is easily identifiable,
- if the radar works but gives incorrect distances (with a certain shift of x meters): for example ($150\ m \ \rightarrow 150 - x\ m$) where x is a constant term, then we cannot detect this fault,
- if the relative velocity and the distance between the vehicles are measured separately by the radar, then we check consistency between them, i.e. at each period (for example 2 seconds) if the variation in the distance corresponds to the variation in the relative velocity. If there is a discrepancy then we conclude that the radar is faulty.

The output of the local algorithm is the radar state (normal/abnormal) s_r:

$$s_r = radar(RD, RV)$$

4) Global diagnoser (or Supervisor): It takes as inputs the outputs of the local diagnosers and its goal is to do some global consistency checking and to merge the local diagnosis decisions in order to obtain one global diagnosis decision and carry out the appropriate recovery action:

$$(c, TS) = global(s_1, s_2, s_3, s_4, s_5, s_6, s_r, V_w, V_t)$$

c and TS represent the control (recovery action) and the truck speed respectively.

Several fault scenarios and recovery actions have been analyzed. The first class of scenarios is composed of wheels sensors failures and/or sensor failure on the rotation speed of the shaft engine (transmission fault). The global diagnosis and recovery actions are described as follows:

- (a) if 1 to 3 wheels sensors are faulty out of the 6 (determined by the wheels diagnoser), the recovery action consists for the SDK function in using the truck speed V_w calculated as average values provided by the correct sensors (between 3 and 5). The global diagnoser compares V_t to V_w and, in case of discrepancy, concludes also to a faulty transmission sensor,
- (b) if 4 wheels sensors are faulty (determined by the wheels diagnoser), a comparison between the speed V_w computed from the two ones assumed to be correct and V_t is performed by the global diagnoser. In case of consistency, the recovery action decided by the global diagnoser consists in using the truck speed calculated as average values provided by the correct wheel sensors and the transmission sensor (3 out of 7),
- (c) in all other cases, i.e. when at most 2 sensors out of 7 provide consistent measurements, the recovery action decided by the global diagnoser consists in disabling the SDK function.

The second scenarios category includes the radar failure. It may pass unnoticed, in particular if RD and RV measurements are not independent, because we cannot provide analytical redundancies between the truck and the front vehicle. In absence of such redundancy, the only check we can do is to verify that the behavior of the front vehicle, in terms of velocity and acceleration, as deduced by the global diagnoser from the truck's behavior (TS) and the radar measurement, is not physically impossible, i.e. does not violate the laws of dynamics. In case of physical impossibility detected, then the recovery action taken by the global diagnoser consists in disabling the SDK controller.

Obviously both scenarios can combine, allowing multiple fault diagnosis of the wheels and transmission sensors and of the radar.

III. DIAGNOSIS DEPLOYMENT

In this section, we propose a deployment solution of the decentralized model-based fault diagnosis strategy (see figure 4) in the electronic architecture of a Renault Truck's vehicle. The SDK electronic architecture part is composed of three Electronic Control Units (ECU) that communicate between them through a bus topology. The ECUs exchange messages that follow the Control Area Network (CAN) protocol, which is low-cost and very wide-spread in the road transport domain. The three ECUs (ECU1, 2 and 3) are respectively linked to wheels, transmission and radar (+SDK algorithm) control functions. We also assume that there is at least another ECU (ECU4) in the vehicle for other high level control functions, which is a very likely situation in modern vehicles. ECU4 will be used for embedding the global diagnoser and recovery functions.

We first list the manufacturing constraints that motivate the adopted deployment solution. We next present the classical on-board diagnosis techniques in vehicles. Finally, we describe the principle of our on-board diagnosis strategy, based on the above electronic architecture. The next section will present an integration and validation platform.

A. Motivations

From an end user point of view, breakdowns and misfunctioning can lead to a loss of safety for the driver and his environment - a major breakdown which necessitates an emergency stop and repair - or a company's performance penalty, due to the need of a vehicle maintenance. From the vehicle manufacturer point of view, these risks must be avoided because they can mostly cause a loss of corporate image and in-field yield.

Beside that, the road transport industry is a very competitive market and the integration of innovative features, such as diagnosis, is highly driven by economical considerations. That is why most of control organs of modern fuel vehicles are embedded and architectured around a limited number of ECUs and communication buses. This evolution towards a very wide use of ECUs was also pushed by the need to reach new challenges such as environmental, performance, security and driving assistance requirements.

As a result, the following four major requirements have driven our diagnosis deployment strategy:

- the number of ECUs has to remain unchanged,
- the number of buses has to remain unchanged and the same communication protocol should be used if possible,
- the additional bus load related to diagnosis information must not affect the current real time performance,
- the diagnosis procedure must be achieved within a bounded time.

Note that, here, we do not address the problem of ECUs load sharing and balancing between the control functions already embedded in them and the diagnosis control ones that will be embedded. This issue will be addressed in a future paper. Nevertheless, the response time of a control function mainly depends on the total communication delay along the bus lines.

Due to the robustness of the CAN-based bus technology, it is a good candidate for enabling the deployment of a decentralized diagnosis strategy. The electronic architecture remains unchanged (no additional ECU and bus). In order to achieve the two last requirements, we developed a SW based diagnosis service that is a SW middleware built on the top of any ECU operating system. It enables the communication of diagnosis information between the local diagnosers and the global diagnoser (as shown in 4) over a loaded CAN bus. It also processes any alert with a bounded time whatever the bus load. It enables the message exchanging with a bounded time while the real time requirements are verified (no more than 2% of delay time) whatever the bus load.

B. Classical diagnosis approaches

On-board Vehicle diagnosis (OBD) refers to vehicle self-detection, localization, identification and reporting capabilities [10], [11]. Early OBD versions for fuel vehicle managed by electronic simply switched-on a malfunction indicator light in the vehicle if any problem was detected. The diagnosis was next performed by an operator in a garage with the aid of a terminal connected to the vehicle electronic. Efficient diagnosis tools were developped for tracking the problem such as [12].

Modern OBD provides realtime diagnosis data in addition to standardized diagnosis trouble codes (DTC) which rapidly allow the vehicle to self-identify and possibly, self-repair by itself the problem during the driving. Otherwise, some vehicles activate a downgraded mode that allows the vehicle driving in safe conditions even in the presence of problems [13]. In parallel, the vehicle switches-on a driver indicator light that points the need of an emergency maintenance (the driver must reach the closest garage) or stop.

Current SAE[1] and ISO[2] standards specify the hardware (connector, network) and communication protocol (Open System Interconnection model) for exchanging diagnostic data over the ECUs and external terminals. Some engineering companies propose tools that allow the ECU original equipment manufacturers (OEM) implementing the standardized DTC and customer-specific diagnosis requirements in their ECU, such as [14] (diagnostic-oriented process flow).

[1]Society of Automobile Engineers
[2]International Standardization Organization

C. Deployment of the decentralized diagnosis

The decentralized diagnosis system, described in section II, is deployed in an electronic architecture of four ECUs which communicate over a single CAN based bus.

The local diagnosers only transmit boolean signals, which take two states: *normal* or *abnormal*, to the global diagnoser. Note that the local diagnosers do not exchange directly diagnosis information with each other. A local diagnoser outputs a *normal* state whenever no fault is detected. During this *normal* state, no additional bus load is due to the diagnosis protocol. Conversely, an *abnormal* state indicates that a problem has likely occurred in a sensor and when it appears, the local diagnoser must immediately send diagnosis information to the remote global diagnoser in order to compute a global diagnosis and, if the need arises, to perform a recovery procedure.

For enabling this event-based procedure, we insert a Local Diagnosis Service (LDS) in every ECU that embeds a local diagnoser. A LDS reads the outputs of the local diagnoser (*normal*/*abnormal*). When an event '*normal* $\rightarrow$ *abnormal*' occurs, it virtually creates a high priority communication channel between the local diagnoser and the global diagnoser, so that the former can immediately transmit the diagnosis information to the latter within a bounded time. The diagnosis information is embedded in CAN messages. In addition, a CAN message contains a control header that especially defines the transmitter and receiver identifiers and the priority level of the communication. On the opposite side, a Supervision Diagnosis Service (SDS) is inserted in the global diagnoser ECU. It monitors the bus load and gathers the diagnosis information sent by any LDS. It next triggers the global diagnoser. Note that the LDS and SDS can periodically check if respectively the SDS or any LDS is safe and ready.

Figure 5 illustrates the different behavior phases of both LDS and SDS before and after the occurrence of a problem.

Figure 5. Decentralized diagnosis protocol over a CAN bus

Initialisation phase: The first phase consists in the establishment of the list of available LDSs. Each LDS sends to the SDS a specific CAN message for initialization which includes its identification number. When the SDS has received the initialization message from all LDSs, the protocol enters the monitoring phase. This instant is materialized by the broadcasting of a specific message from the SDS to all LDSs.

Monitoring phase: During this phase, no messages are exchanged between the LDSs and the SDS. This phase corresponds to a situation where the system is operating normally, without local diagnosis event from the local diagnosers. This implies that there is not any over traffic due to the diagnosis during the normal operation of the system.

Alert phase: This phase begins when an abnormal event is detected by a local diagnoser. At this moment, the LDS of the concerned ECU sends a specific alert event message to the SDS. The instant of the first alert event message emission materializes the beginning of a period when the SDS is waiting for other alert event messages that would complete the information for the global diagnosis. At the end of this period, the SDS gives the order to the global diagnoser to compute the global diagnosis. When this is done, the SDS broadcasts a specific message to all LDSs for entering again in the monitoring phase.

Vivacity check: While the protocol is in monitoring phase, the SDS can initiate a vivacity check: the SDS broadcasts a specific CAN message to all LDSs. When the LDSs receive this message during their monitoring phase, they send an acknowledgement message for proving that the on-board diagnosis is correctly running.

IV. EVALUATION OF THE DIAGNOSIS

A. Matlab/Simulink Simulation

We developed a physical simulation model of the SDK environment in MATLAB/Simulink. We adopted a component based modeling paradigm, where parameterized simulation models of generic components (SDK controller, radar, wheels, transmission, engine, and supervisor) were developed within a component library. The different local models are constructed by instantiating different components from the library, specifying their parameters, and connecting the components to each other in the appropriate fashion.

Except the supervisor, each component model includes its associated fault modes. The fault mode, time of fault injection, and fault magnitude (where applicable) can all be specified. In general, each fault mode is mapped to a change in component mode and a fault-dependent magnitude parameter. Because each fault mode is parameterized within the Simulink model, a fault can be injected programmatically (i.e., the fault mode, injection time, and magnitude are specified) either at the beginning of the simulation, or while the simulation is running.

B. Prototypes (Hardware-In-the-Loop (HIL) Simulation)

The proposed HIL scheme is given in figure 6. It shows

Figure 6. General diagram of the prototype of embedded diagnosis of the SDK function

that the electro-mechanical subsystems (Truck model) and local diagnosers are emulated (simulated in real time) and

the supervisor is real. The main softwares used are MATLAB/Simulink and CANoe[3], while the hardware entity is a physical card including the supervisor and a HMI (Human-Machine Interface).

Simulating in real time the ECUs network, observing the CAN-bus load, and determining the necessary performance of the hardware which is being developed are the main reasons to use CANoe.

1) CANoe and Matlab/Simulink Interface: This interface allows execution of Simulink models inside the CANoe network simulation environment. It offers various paths for data exchange between CANoe and MATLAB (see figure 7 (a)). The CANoe simulation and the Simulink models can communicate directly through a signal interface or through CANoe environment and system variables.

Figure 7. (a) Data exchange between CANoe and MATLAB/Simulink, (b) Redirection of a Simulink signal (output of the wheels diagnoser) to a CANoe signal and CANoe environment variable

In the first step, the simulation of the SDK environment with its control and its diagnosers has been implemented and validated with Matlab/Simulink. Then, the interfacing stage of the local diagnosers with CANoe has been performed. The difficulty of this stage is to translate continuous Simulink signals to events in order to activate the treatments and generate messages in CANoe. To this end, some links of the Matlab/Simulink simulator have been cut off and connected to CANoe communication blocks. So, CANoe and the Simulink models may communicate via signals and environment variables[4]. These are computed from Simulink signals (see figure 7 (b)). In our case, the CANoe signal has been configured so that its change does not activate sending CAN message automatically. It also depends on the environment variables evolution.

As shown in figure 6, the CANoe environment is connected to the physical card through a real CAN-bus. That is why this interface has been operated in Hardware-In-the-Loop (HIL) mode and the simulation has been run in the CANoe execution environment. So, the local ECUs have been simulated simultaneously within a single CANoe environment. With the Simulink Real-Time Workshop one DLL has been generated per network node (or local ECU) and then loaded in CANoe's simulation environment. We can then dispense with Matlab/Simulink and run in real-time emulation to interact with real platforms via the real CAN-bus.

[3]CANoe is an all-round tool of Vector Informatik company for the development, testing and analysis of entire ECU networks and individual ECUs.

[4]A signal is destined to be transported in a CAN-bus message, while a variable environment does not have this vocation.

2) Supervisor: It is implemented on a physical platform (see figure 8 (a)) that is an electronic card, DIAFORE-card, interfaced with the local ECUs via a real CAN-bus, developed by Serma Ingénierie company. It is developed around 32-bits RISC micro-controller. An RS232 interface and a FPGA (Field-Programmable Gate Array) card are implemented on this DIAFORE-card.

Figure 8. (a) DIAFORE-card including the diagnosis supervisor, (b) Screenshot of the HMI that indicates the diagnostic messages sent to the supervisor by the local diagnosers

The global diagnosis function (or algorithm) has been implemented in C language and then integrated with the "management engine of the CAN-bus messages" on the DIAFORE-card. The supervisor's role is to treat all meaningful CAN messages and retrieve the necessary informations to perform the global diagnosis. The counter-actions associated to the verdict of the global diagnosis are then sent to the concerned other ECUs through the Can-bus network.

To visualize the diagnosis results and the supervisor activity in a computer linked to the DIAFORE card by a RS232 link, a Labview HMI has been developed (see figure 8 (b)).

By using the FPGA card the supervisor can also perform self-test queries on the local ECUs. The query response of each local ECU is a CAN-bus trame containing several types of informations about their state. These informations are then sent to the HMI by the supervisor in order to be visualized.

C. Use cases

This study aims at validating the communication behavior between the local diagnosers and the global one (supervisor) under a loaded CAN-bus constraint. The diagnosis latency is then evaluated to estimate its determinism under some varying conditions of load and relative priority in relation to the existing CAN messages. The diagnosis protocol is always performed with a nominal latency when the diagnostic messages are configured with highest priority. Conversely, if there are higher priority messages, then it depends on the load on the CAN-bus.

D. Evaluation

While the mean period of the broadcasting messages of the support is greater than or equal to 10 ms, it was observed that the diagnosis latency remains nominal. When the mean period of the broadcasting messages of the support has a value less than 10 ms, then the diagnosis latency increases on average and is even infinite in some cases. However, the normal load of a CAN-bus of a truck is well under that threshold.

While the demonstrators have proven the feasibility and utility of model-based diagnosis for on-board diagnosis,

more substantial work is needed in order to promote the application of the technology for a real vehicle:

- Firstly, the identifiers of diagnosis messages should be adapted in order to coexist with identifiers of other CAN messages. Indeed, the choice of identifiers defines not only the content but also the message priority on the bus. Theoretically, it has an impact on the overall functioning. The results obtained by the validation phase of the decentralized algorithm allow us to be optimistic on this point.
- Furthermore, in a real truck vehicle, the SDK function must coexist with other distributed functions. That is why, various parameters must be adjusted as the diagnosis latency.

V. CONCLUSIONS

In this paper we have developed a diagnosis algorithm of the SDK system and a software architecture in order to board it inside truck vehicle in a decentralized manner. The model-based diagnosis approach has been used because it presents the advantage that no prior knowledge of possible faults or symptoms is needed. It relies only on a given model of the correct functioning of the system and proceeds by comparing the behaviors of the model and of the actual system (as known through the observations given by sensors).

The originality of the accomplished work is based on two contributions. The first one is the distribution of diagnosis algorithms on several ECUs by using the decentralized diagnosis approach (the method has only been applied, in the practical context of industrial applications, in a centralized manner). This approach uses a set of diagnosers. Each diagnoser observes a part of the SDK system and takes a local decision about the occurrence of a fault and its localisation. The construction of the local diagnosers is based on a modular modeling of the plant elements. All local diagnosers decisions must be merged by a dedicated supervisor in order to obtain one global diagnosis decision and to take also any recovery action. This fusion can be realized by a coordinator. The second contribution is the design and deployment of the decentralized model-based fault diagnosis approach for the SDK system. The attention was paid to minimize the additional traffic generated by the diagnosis function and to respect the real application constraints (performance, diagnosis latency, etc.). An on-board diagnosis using Hardware-In-the-Loop scheme under specifications (contraints) near to a truck "Renault Trucks" has been performed.

The decentralized embedded diagnosis for the SDK system inside real vehicles is mature from the point of view of research and of feasibility. However, we should extend and develop algorithms robustness, take into account the protocols and details of the hardware architecture and existing software, conduct tests on many scenarios, and measure in situ the quality (correctness, precision, time delay) of the diagnosis and its compatibility with existing functions (induced traffic on the CAN-bus, transparency). Moreover, in order to verify a priori that the set of local diagnosers and supervisor is capable of diagnosing a given set of faults within a bounded delay, a notion of diagnosability must be studied.

VI. ACKNOWLEDGMENTS

The work reported in this paper was supported by the project DIAFORE (Diagnosis for Distributed Functions) which is funded by the French National Research Agency (ANR) and SYSTEM@TIC PARIS-REGION Cluster, with the support of French Environment and Energy Management Agency (ADEME) and French Program of Research, Experimentation and Innovation in Land transport (PREDIT). Many thanks to industrial partners (RENAULT TRUCKS/VOLVO SAS and SERMA INGENIERIE) in this project for their support upon completion of the simulation platform.

REFERENCES

[1] Davis, L.C., Effect of Adaptive Cruise Control Systems on Traffic Flow. Physical Review E, 69(6), 2004.

[2] Darkhovski B. and Staroswiecki M., Theoretic Approach to Decision in FDI. IEEE Transactions On Automatic Control, 48(5), 2003.

[3] Patten R.J., Frank P.M. and Clark R., Issues of fault diagnosis for dynamic systems. Springer, London, 2000.

[4] Chow E. and Willsky A., Analytic redundancy and the design of robust failure detection systems. IEEE Transactions on Automatic Control, 29, 603-614, 1984.

[5] Shraim H., Nasri O., Dague P., Héron O. and Cartron M. , Smart Distance Keeping: Modeling and Perspectives for Embedded Diagnosis. In ISMS2010: 1st International Conference on Intelligent Systems, Modelling and Simulation, Liverpool, England, 2010.

[6] Peysson F., Noura H. and Younes R., Diagnostic de défauts sur un moteur diesel. CIFA06, Bordeaux, France, 2006.

[7] Sampath M., Sengupta R., Lafortune S., Sinnamohideen K. and Teneketzis D., Diagnosability of discrete event systems. In 11th International Conference on Analysis and Optimization of Systems, Sophia-Antipolis, France,1994.

[8] Wang Y., Yoo T.S. and Lafortune S., New Results on Decentralized Diagnosis of Discrete Event Systems. Annual Allerton Conference on Communication, Control and Computing, 2004.

[9] Qiu W., Decentralized / distributed failure diagnosis and supervisory control of discrete event systems. PhD thesis, Iowa State University, 2005.

[10] O'Reilly P., An overview of the potential contribution of diagnostics to improving vehicle safety and reducing vehicle emissions. IEE Colloquium on Vehicle Diagnostics in Europe, 1/1-1/12, 1994.

[11] Greening P., On-board diagnostics for control of vehicle emissions. IEEE Colloquium on Vehicle Diagnostics in Europe, 5/1-5/6, 1994.

[12] Ressencourt H., Diagnostic hors-ligne à base de modèles : approche multi-modèle pour la génération automatique de séquences de tests, application au domaine de l'automobile. PhD thesis, 2008.

[13] Fromion A., Apparatus for the differential transmission of information between at least two devices in a vehicle, European patent EP19960401324, 2003.

[14] Frank H. and Schmidts U., Vehicle Diagnostics - The whole Story. Vector Informatik, Press article, www.vector.com, 2007.